Self-Taught Hashing for Fast Similarity Search

Dell Zhang
DCSIS
Birkbeck, University of London
Malet Street
London WC1E 7HX, UK
dell.z@ieee.org

Jun Wang
Dept of Computer Science
University College London
Gower Street
London WC1E 6BT, UK
jun.wang@cs.ucl.ac.uk

Deng Cai
State Key Lab of CAD&CG
Zhejiang University
100 Zijinggang Road
Hangzhou 310058, China
dengcai@cad.zju.edu.cn

Jinsong Lu
DEMS
Birkbeck, University of London
Malet Street
London WC1E 7HX, UK
jingsong.lu@gmail.com

ABSTRACT

The ability of fast similarity search at large scale is of great importance to many Information Retrieval (IR) applications. A promising way to accelerate similarity search is semantic hashing which designs compact binary codes for a large number of documents so that semantically similar documents are mapped to similar codes (within a short Hamming distance). Although some recently proposed techniques are able to generate high-quality codes for documents known in advance, obtaining the codes for previously unseen documents remains to be a very challenging problem. In this paper, we emphasise this issue and propose a novel Self-Taught Hashing (STH) approach to semantic hashing: we first find the optimal l-bit binary codes for all documents in the given corpus via unsupervised learning, and then train l classifiers via supervised learning to predict the l-bit code for any query document unseen before. Our experiments on three real-world text datasets show that the proposed approach using binarised Laplacian Eigenmap (LapEig) and linear Support Vector Machine (SVM) outperforms state-of-the-art techniques significantly.

Categories and Subject Descriptors

H.2.8 [**Database Management**]: Database Applications—*data mining*; H.3.1 [**Information Storage and Retrieval**]: Content Analysis and Indexing; H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval; I.2.6 [**Artificial Intelligence**]: Learning; I.5.2 [**Pattern Recognition**]: Design Methodology—*classifier design and evaluation*

General Terms

Algorithms, Experimentation, Performance

Keywords

Similarity Search, Semantic Hashing, Laplacian Eigenmap, Support Vector Machine.

SIGIR'10, July 19–23, 2010, Geneva, Switzerland.

1. INTRODUCTION

The problem of *similarity search* (aka *nearest neighbour search*) is: given a query document[1], find its most similar documents from a very large document collection (corpus). It is of great importance to many Information Retrieval (IR) [30] applications, such as near-duplicate detection [18], plagiarism analysis [43], collaborative filtering [26], caching [32], and content-based multimedia retrieval [28].

Recently, with the rapid evolution of the Internet and the increased amounts of data to be processed, how to conduct fast similarity search at large scale has become an urgent research issue. A promising way to accelerate similarity search is *semantic hashing* [34] which designs compact binary codes for a large number of documents so that semantically similar documents are mapped to similar codes (within a short Hamming distance). It is extremely fast to perform similarity search over such binary codes [42], because

- the encoded data are highly compressed and thus can be loaded into the main memory;
- the Hamming distance between two binary codes can be computed efficiently by using bit XOR operation and counting the number of set bits [25, 46]: an ordinary PC today would be able to do millions of Hamming distance computation in just a few milliseconds.

Furthermore, we usually just need to retrieve a small number of the most similar documents (i.e., nearest neighbours) for a given query document rather than computing its similarity to all documents in the collection. In such situations, we can simply return all the documents that are hashed into a tight

[1]In similarity search, a document is used as the query for retrieval, which is fundamentally different with the standard keyword search paradigm, e.g., in TREC.

Hamming ball centred around the binary code of the query document. For example, assuming that we use 4-bit binary codes, if the query document is represented as '0000', then we can just check this code as well as those 4 codes within one Hamming distance to it (i.e., having one bit difference with it) — '1000', '0100', '0010', and '0001' — and return the associated documents back. It will also be easy to filter or re-rank the very small set of "good" documents (returned by semantic hashing) based on their full content, so as to further improve the retrieval effectiveness with just a little extra time [42].

In addition, similarity search serves as the basis of a classic non-parametric machine learning method, the k-Nearest-Neighbours (kNN) algorithm [31], for automated text categorisation [37] and so on. By enabling fast similarity search at large scale, semantic hashing makes it feasible to exploit "the unreasonable effectiveness of data" [14] to accomplish traditionally difficult tasks. For example, researchers recently achieved great success in scene completion and scene recognition using millions of images on the Web as training data [15, 44].

Although some recently proposed techniques are able to generate high-quality codes for the documents known in advance, obtaining the codes for previously unseen documents remains to be a very challenging problem [42]. Existing methods either have prohibitively high computational complexity or impose exceedingly restrictive assumptions about data distribution (see Section 3.2). In this paper, we emphasise this issue and propose a novel Self-Taught Hashing (STH) approach to semantic hashing. As illustrated in Figure 1, we first find the optimal l-bit binary codes for all documents in the given corpus via unsupervised learning, and then train l classifiers via supervised learning to predict the l-bit code for any query document unseen before.

Our experiments on three real-world text datasets show that the proposed approach using binarised Laplacian Eigenmap (LapEig) [3] and linear Support Vector Machine (SVM) [23, 36] outperforms state-of-the-art techniques significantly, while maintaining a high running speed.

The rest of this paper is organised as follows. In Section 2, we review the related work. In Section 3, we present our approach in details. In Section 4, we show the experimental results. In Section 5, we make conclusions.

2. RELATED WORK

There has been extensive research on fast similarity search due to its central importance in many applications. For a low-dimensional feature space, similarity search can be carried out efficiently with pre-built space-partitioning index structures (such as KD-tree) or data-partitioning index structures (such as R-tree) [7]. However, when the dimensionality of feature space is high (say > 10), similarity search aiming to return exact results cannot be done better than the naive method — a linear scan of the entire collection [45]. In the IR domain, documents are typically represented as feature vectors in a space of more than thousands of dimensions [30]. Nevertheless, if the complete exactness of results is not really necessary, similarity search in a high-dimensional space can be dramatically speeded up by using hash-based methods which are purposefully designed to approximately answer queries in virtually constant time [42].

Such hash-based methods for fast similarity search can be considered as a means for embedding high-dimensional feature vectors to a low-dimensional Hamming space (the set of all 2^l binary strings of length l), while retaining as much as possible the semantic similarity structure of data. Unlike standard dimensionality reduction techniques such as Latent Semantic Indexing (LSI) [5, 8] and Locality-Preserving Indexing (LPI) [17, 16], hashing techniques map feature vectors to *binary* codes, which is key to extremely fast similarity search (see Section 1). One possible way to get binary codes for text documents is to binarise the real-valued low-dimensional vectors (obtained from dimensionality reduction techniques like LSI) via thresholding [34]. An improvement on binarised-LSI that directly optimises a Hamming distance based objective function, namely Laplacian Co-Hashing (LCH), has been proposed recently [50].

The most well-known hashing technique that preserves similarity information is probably Locality-Sensitive Hashing (LSH) [1]. LSH simply employs random linear projections (followed by random thresholding) to map data points close in a Euclidean space to similar codes. It is theoretically guaranteed that as the code length increases, the Hamming distance between two codes will asymptotically approach the Euclidean distance between their corresponding data points. However, since the design of hash functions for LSH is *data-oblivious*, LSH may lead to quite inefficient (long) codes in practice [34, 48].

Several recently proposed hashing techniques attempt to overcome this problem by finding good *data-aware* hash functions through machine learning. In [34], the authors proposed to use stacked Restricted Boltzmann Machine (RBM) [19, 20], and showed that it was indeed able to generate compact binary codes to accelerate document retrieval. Researchers have also tried the boosting approach to Similarity Sensitive Coding (SSC) [38] and Forgiving Hashing (FgH) [2] — they first train AdaBoost [35] classifiers with similar pairs of items as positive examples (and also non-similar pairs of items as negative examples in SCC), and then take the output of all (decision stump) weak learners on a given document as its binary code. In [44], both stacked-RBM and boosting-SSC were found to work significantly better and faster than LSH when applied to a database containing tens of millions of images. In [48], a new technique called Spectral Hashing (SpH) was proposed. It has demonstrated significant improvements over LSH, stacked-RBM and boosting-SSC in terms of the number of bits required to find good similar items. There is some resemblance between the first step of SpH and the unsupervised learning stage of our STH approach, because both are related to spectral graph partitioning [6, 13, 40]. Nevertheless, we use a different spectral method and take a different way to address the *entropy maximising* criterion (see Section 3.1). More importantly, in order to process query documents, SpH has to assume that the data are uniformly distributed in a hyper-rectangle, which is apparently very restrictive. In contrast, our proposed STH approach can work with any data distribution and it is much more flexible (see Section 3.2). The superiority of STH to SpH has been confirmed by our experimental results (see Section 4).

A somewhat related, but different, line of research is to use hashing representations for machine learning [41, 47]. The objective of such techniques is to accelerate complex learning algorithms, but not similarity search. Our work is basically the other way around.

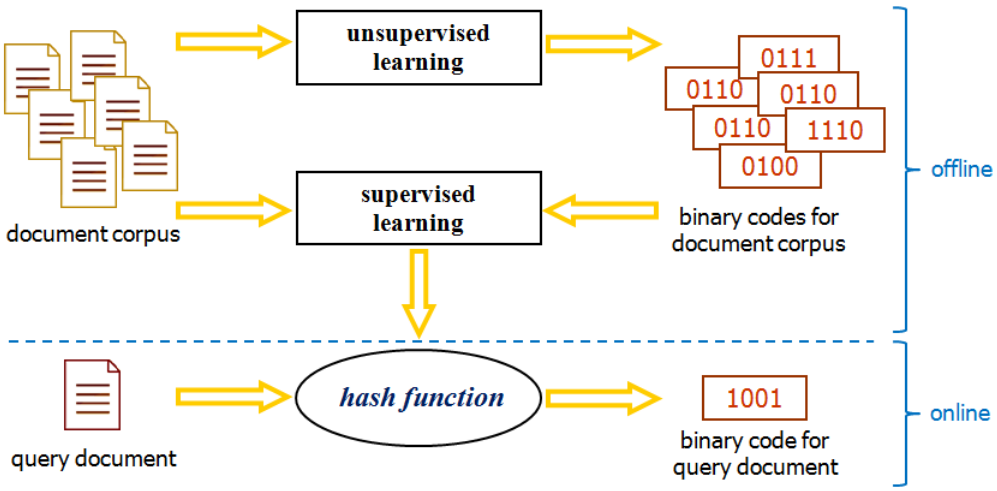


Figure 1: The proposed STH approach to semantic hashing.

3. APPROACH

The proposed Self-Taught Hashing (STH) approach to semantic hashing is a *general* learning framework that consists of two distinct stages, as illustrated in Figure 1. We call the approach "self-taught" because the hash function is learnt from the data that are auto-labelled by itself in the previous stage[2].

3.1 Stage 1: Unsupervised Learning of Binary Codes

Given a collection of n documents which are represented as m-dimensional vectors $\{\mathbf{x}_i\}_{i=1}^n \subset \mathbb{R}^m$. Let X denote the $m \times n$ term-document matrix: $[\mathbf{x}_1, \ldots, \mathbf{x}_n]$. Suppose that the desired length of code is l bits. We use $\mathbf{y}_i \in \{-1, +1\}^l$ to represent the binary code for document vector $\mathbf{x}_i$, where the p-th element of $\mathbf{y}_i$, i.e., $y_i^{(p)}$, is $+1$ if the p-th bit of code is on, or -1 otherwise. Let Y denote the $n \times l$ matrix whose i-th row is the code for the i-th document, i.e., $[\mathbf{y}_1, \ldots, \mathbf{y}_n]^T$.

A "good" semantic hashing should be ***similarity preserving*** to ensure effectiveness. That is to say, semantically similar documents should be mapped to similar codes within a short Hamming distance.

Unlike the existing approaches (such as SpH [48]) that aim to preserve the *global* similarity structure of all document pairs, we focus on the *local* similarity structure, i.e., k-nearest-neighbourhood, for each document. Since IR applications usually put emphasis on a small number of most similar documents for a given query document [30], preserving the global similarity structure is not only unnecessary but also likely to be sub-optimal for our problem. Therefore, using the *cosine similarity*[3] [30], we construct our $n \times n$ *local* similarity matrix W as

$$W_{ij} = \begin{cases} \frac{\mathbf{x}_i^T \mathbf{x}_j}{\|\mathbf{x}_i\| \cdot \|\mathbf{x}_j\|} & \text{if } \mathbf{x}_i \in N_k(\mathbf{x}_j) \text{ or } \mathbf{x}_j \in N_k(\mathbf{x}_i) \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

where $N_k(\mathbf{x})$ represents the set of k-nearest-neighbours of document $\mathbf{x}$. In other words, W is the adjacency matrix of the k-nearest-neighbours graph for the given corpus [3]. A by-product of focusing on such a local similarity structure instead of the global one is that W becomes a sparse matrix. This not only leads to much lower storage overhead, but also brings a significant reduction to the computational complexity of subsequent operations. Furthermore, we introduce a diagonal $n \times n$ matrix D whose entries are given by $D_{ii} = \sum_{j=1}^n W_{ij}$. The matrix D provides a natural measure of document importance: the bigger the value of D_{ii} is, the more "important" is the document $\mathbf{x}_i$ as its neighbours are strongly connected to it [3].

The Hamming distance between two binary codes $\mathbf{y}_i$ and $\mathbf{y}_j$ (corresponding to documents $\mathbf{x}_i$ and $\mathbf{x}_j$) is given by the number of bits that are different between them, which can be calculated as $\frac{1}{4}\|\mathbf{y}_i - \mathbf{y}_j\|^2$. To meet the *similarity preserving* criterion, we seek to minimise the weighted average Hamming distance (as in SpH [48])

$$\frac{1}{4}\sum_{i=1}^n \sum_{j=1}^n W_{ij}\|\mathbf{y}_i - \mathbf{y}_j\|^2 \quad (2)$$

because it incurs a heavy penalty if two similar documents are mapped far apart. After some simple mathematical transformation, the above objective function can be rewritten in matrix form as $\frac{1}{4}\mathrm{Tr}(Y^T L Y)$, where $L = D - W$ is the *graph Laplacian* [6], and $\mathrm{Tr}(\cdot)$ means the matrix trace.

We found the above objective function (2) actually proportional to that of a well-known manifold learning algorithm, Laplacian Eigenmap (LapEig) [3], except that LapEig does not have the constraint $\mathbf{y}_i \in \{-1, +1\}^l$. So, if we relax this discreteness condition but just keep the *similarity preserving* requirement, we can get the optimal l-dimensional real-valued vector $\tilde{\mathbf{y}}_i$ to represent each document $\mathbf{x}_i$ by solv-

[2]It is, however, worth noticing that the term "self-taught learning" has been mentioned in [33] where the intention was to describe a strategy for transfer learning based on sparse coding, whereas in this paper the term has a rather different meaning.

[3]Our approach can work with any legitimate similarity measure, though we focus on cosine similarity in this paper.

ing the following LapEig problem:

$$\begin{aligned} \arg\min_{\tilde{Y}} \quad & \mathrm{Tr}(\tilde{Y}^T L \tilde{Y}) \\ \text{subject to} \quad & \tilde{Y}^T D \tilde{Y} = I \\ & \tilde{Y}^T D \mathbf{1} = \mathbf{0} \end{aligned} \tag{3}$$

where $\mathrm{Tr}(\tilde{Y}^T L \tilde{Y})$ gives the real relaxation of the weighted average Hamming distance $\mathrm{Tr}(Y^T L Y)$, and the two constraints prevent the collapse into a subspace of dimension less than l. The solution of this optimisation problem is given by $\tilde{Y} = [\mathbf{v}_1, \ldots, \mathbf{v}_l]$ whose columns are the l eigenvectors corresponding to the smallest eigenvalues of the following generalised eigenvalue problem (except the trivial eigenvalue 0):

$$L\mathbf{v} = \lambda D \mathbf{v} \tag{4}$$

The above LapEig formulation (3) may look similar to the first step of SpH [48]. This is because SpH is motivated by a spectral graph partitioning method *ratio-cut* [13], while LapEig is closely connected to another spectral graph partitioning method *normalised-cut* [40]. Many independent studies have shown that normalised-cut has better theoretical properties and empirical performances than ratio-cut [6, 40].

We now convert the above l-dimensional real-valued vectors $\tilde{\mathbf{y}}_1, \ldots, \tilde{\mathbf{y}}_n$ into binary codes via thresholding: if the p-th element of $\tilde{\mathbf{y}}_i$ is larger than the specified threshold, $y_i^{(p)} = +1$ (i.e., the p-th bit of the i-th code is on); otherwise, $y_i^{(p)} = -1$ (i.e., the p-th bit of the i-th code is off).

A "good" semantic hashing should also be ***entropy maximising*** to ensure efficiency, as pointed out by [2]. According to the *information theory* [39]: the maximal entropy of a source alphabet is attained by having a uniform probability distribution. If the entropy of codes over the corpus is small, it means that documents are mapped to only a small number of codes (hash bins), thereby rendering the hash table inefficient. To meet this *entropy maximising* criterion, we set the threshold for binarising $\tilde{y}_1^{(p)}, \ldots, \tilde{y}_n^{(p)}$ to be the *median* value of $\mathbf{v}_p$. In this way, the p-th bit will be on for half of the corpus and off for the other half. Furthermore, as the eigenvectors $\mathbf{v}_1, \ldots, \mathbf{v}_l$ given by LapEig are orthogonal to each other, different bits $y^{(1)}, \ldots, y^{(l)}$ in the generated binary codes will be uncorrelated. Therefore this thresholding method gives each distinct binary code roughly equal probability of occurring in the document collection, thus achieves the best utilisation of the hash table.

3.2 Stage 2: Supervised Learning of Hash Function

Mapping all documents in the given corpus to binary codes does not completely solve the problem of semantic hashing, because we also need to know how to obtain the binary codes for query documents, i.e., new documents that are unseen before. This problem, called *out-of-sample extension* in manifold learning, is often addressed using the Nystrom method [4, 9]. However, calculating the Nystrom extension of a new document is as computationally expensive as an exhaustive similarity search over the corpus (that may contain millions of documents), which makes it impractical for semantic hashing. In LPI [17, 16], LapEig [3] is extended to deal with new samples by approximating a linear function to the embedding of LapEig. However, the computational complexity of LPI is very high because its learning algorithm involves eigen-decompositions of two large dense matrices. It is infeasible to apply LPI if the given training corpus is large. In SpH [48], new samples are handled by utilising the latest results on the convergence of graph Laplacian eigenvectors to the Laplace-Beltrami eigenfunctions of manifolds. It can achieve both fast learning and fast prediction, but it relies on a very restrictive assumption that the data are uniformly distributed in a hyper-rectangle.

Overcoming the limitations of the above techniques [4, 9, 17, 16, 48], this paper proposes a novel method to compute the binary codes for query documents by considering it as a supervised[4] learning problem: we think of each bit $y_i^{(p)} \in \{+1, -1\}$ in the binary code for document $\mathbf{x}_i$ as a binary class label (class-"on" or class-"off") for that document, and train a binary classifier $y^{(p)} = f^{(p)}(\mathbf{x})$ on the given corpus that has already been "labelled" by the above binarised-LapEig method, then we can use the learned binary classifiers $f^{(1)}, \ldots, f^{(l)}$ to predict the l-bit binary code $y^{(1)}, \ldots, y^{(l)}$ for any query document $\mathbf{x}$. As mentioned in the previous section, different bits $y^{(1)}, \ldots, y^{(l)}$ in the generated binary codes are uncorrelated. Hence there is no redundancy among the binary classifiers $f^{(1)}, \ldots, f^{(l)}$, and they can also be trained independently.

In this paper, we choose to use the Support Vector Machine (SVM) [23, 36] algorithm to train these binary classifiers. SVM in its simplest form, linear SVM $f(\mathbf{x}) = \mathrm{sgn}(\mathbf{w}^T \mathbf{x})$ consistently provides state-of-the-art performance for text classification tasks [10, 22, 49]. Given the documents $\mathbf{x}_1, \ldots, \mathbf{x}_n$ together with their *self-taught* binary labels for the p-th bit $y_1^{(p)}, \ldots, y_n^{(p)}$, the corresponding linear SVM can be trained by solving the following quadratic optimisation problem

$$\begin{aligned} \arg\min_{\mathbf{w}, \xi_i \geq 0} \quad & \frac{1}{2}\mathbf{w}^T\mathbf{w} + \frac{C}{n}\sum_{i=1}^{n} \xi_i \\ \text{subject to} \quad & \forall_{i=1}^{n} : y_i^{(p)} \mathbf{w}^T \mathbf{x}_i \geq 1 - \xi_i \end{aligned} \tag{5}$$

A notable advantage of using SVM classifiers here is that we can easily achieve non-linear mappings if necessary by plugging in non-linear kernels [36], though we do not explore this potential in this paper.

3.3 Summary of Approach

We name the above proposed two-stage approach Self-Taught Hashing (STH). In this paper, we choose binarised-LapEig [3] for the unsupervised learning stage and linear-SVM [23, 36] for the supervised learning stage, but obviously it is possible to use other machine learning algorithms.

The *learning* process of STH for a given corpus can be summarized as follows.

1. **unsupervised learning of binary codes**:
   - construct the k-nearest-neighbours graph for the given corpus;
   - embed the documents in an l-dimensional space through LapEig (4) to get an l-dimensional real-valued vector for each document;

[4]Since in the second stage, the supervised learning algorithm uses only the *pseudo*-labels input from the previous unsupervised learning stage, the entire STH approach remains to be unsupervised.

- obtain an l-bit binary code for each document via thresholding the above vectors at their median point, and then take each bit as a binary class label for that document;

2. **supervised learning of hash function**:

   - train l SVM classifiers (5) based on the given corpus that has been "labelled" as above.

Let s denote the average number of non-zero features per document. In the first stage, constructing the k-nearest-neighbours graph takes $O(n^2s+n^2k)$ time using the selection algorithm [7], solving the LapEig problem (4) takes $O(lnkt)$ time using the Lanczos algorithm [12] of t iterations (the value of t is usually quite small), and the median-based binarisation takes $O(ln)$ time again using the selection algorithm [7]. In the second stage, thanks to the recent advances in large-scale optimisation, each of the l linear SVM classifiers can be trained in $O(sn)$ time or even less [24, 21], so all training can be done in $O(lsn)$ time. Both the value of l and the value of k can be regarded as small constants, as usually a short code length is desirable and just a few nearest neighbours are needed. For example, $l \leq 64$ and $k = 25$ in our experiments (see Section 4). Therefore the overall computational complexity of the learning process is roughly quadratic to the number of documents in the corpus while linear to the average size of the documents in the corpus.

The *predicting* process of STH for a given query document is simply to classify the query document using those l learned classifiers and then assemble the output l binary labels into an l-bit binary code. For linear SVM, classifying a document only requires one dot-product operation between two vectors, the aggregated support vector and the document vector (with s' non-zero features), which can be done quickly in $O(s')$ time. Therefore the overall computational complexity of the prediction process for each query document is linear to the size of the query document.

4. EXPERIMENTS

We now empirically evaluate our proposed STH approach (using binarised-LapEig and linear-SVM), and compare its performance with binarised-LSI [34], LCH [50], and SpH [48] that represents the state of the art (see Section 2).

In the following STH experiments, the parameter $k = 25$ when constructing the k-nearest-neighbours graph for LapEig[5], and the SVM implementation is from LIBLINEAR [11] with the default parameter values[6].

4.1 Data

We have conducted experiments on three publicly available real-world text datasets: Reuters21578[7], 20Newsgroups[8] and TDT2[9].

[5]In principle, the value of k for LapEig should be set to the desired number of original nearest neighbours to be retrieved (see Section 4.2).

[6]It is not necessary to fine tune the SVM parameters (such as C) because it has already worked very well with its default parameter values.

[7]http://www.daviddlewis.com/resources/testcollections/reuters21578/

[8]http://people.csail.mit.edu/jrennie/20Newsgroups/

[9]http://www.nist.gov/speech/tests/tdt/tdt98/index.htm

The Reuters21578 corpus is a collection of documents that appeared on Reuters newswire in 1987. It contains 21578 documents in 135 categories. In our experiments, those documents appearing in more than one category were discarded, and only the largest 10 categories were kept, thus leaving us with 7285 documents in total. We use the ModeApte split here which gives 5228 (72%) documents for training and 2057 (28%) documents for testing.

The 20Newsgroups corpus was collected and originally used for document categorisation by Lang [27]. We use the popular 'bydate' version which contains 18846 documents, evenly distributed across 20 categories. The time-based split leads to 11314 (60%) documents for training and 7532 (40%) documents for testing.

The TDT2 (NIST Topic Detection and Tracking) corpus consists of data collected during the first half of 1998 and taken from 6 sources, including 2 newswires (APW, NYT), 2 radio programs (VOA, PRI) and 2 television programs (CNN, ABC). It consists of 11201 on-topic documents which are classified into 96 semantic categories. In our experiments, those documents appearing in more than one category were discarded, and only the largest 30 categories were kept, thus leaving us with 9394 documents in total. We randomly selected 5597 (60%) documents for training and 3797 (40%) documents for testing. The averaged performance based on 10 such random selections is reported in this paper.

All the above datasets have been pre-processed by stopword removal, Porter stemming, and TF-IDF weighting [30].

For the purpose of reproducibility, we shall make the datasets and code used in our experiments publicly available at the first author's homepage upon paper publication.

4.2 Evaluation

Given a dataset, we use each document in the test set as a query to retrieve documents in the training set within a specified Hamming distance, and then compute standard retrieval performance measures: *precision*, *recall*, and their harmonic mean (F_1 measure) [30].

$$precision = \frac{\text{the number of retrieved relevant documents}}{\text{the number of all retrieved documents}} \quad (6)$$

$$recall = \frac{\text{the number of retrieved relevant documents}}{\text{the number of all relevant documents}} \quad (7)$$

The reported performance scores in the following Section are averaged over all test queries in the dataset.

To determine whether a retrieved document is "relevant" to the given query document, we adopt the following two evaluation methodologies:

1. **retrieving original nearest neighbours** — the k most similar documents, i.e., nearest neighbours, in the original vector space are considered as the ground-truth relevant documents ($k = 25$ in our experiments);

2. **retrieving same-topic documents** — the documents on the same topic, i.e., in the same category, are considered as the ground-truth relevant documents.

The former methodology is used in [48][10], while the latter methodology is used in [34]. In our opinion, these two

[10]Actually only precision is used in [48], which is appropriate

methodologies emphasise different aspects of semantic hashing, and thus are suitable for different target IR applications. Therefore we use both of them in this paper.

The absolute performance scores of STH are not as important as how they compare with those of other semantic hashing techniques. As previously mentioned in Section 1, if necessary, we can always spend a little extra time to filter or re-rank the similarity search results based on their full content, thus achieve higher performance scores [42].

4.3 Results

Figure 2 and Figure 3 show the F_1 measure of STH for retrieving original nearest neighbours and same-topic documents respectively[11]. We vary the code length from 4-bit to 64-bit and also the Hamming ball radius (i.e., the maximum Hamming distance between any retrieved document and the query document) from 0 to 3, in order to show their influences on the retrieval performance. It can be seen that when the code length increases, STH is able to achieve a higher F_1 measure (using a bigger Hamming ball radius). However, longer binary codes demand more memory and a bigger Hamming ball radius requires more computation. The optimal trade-off between effectiveness and efficiency can be found by using a validation set of query documents.

Figure 4 and Figure 5 compare STH with several other typical semantic hashing methods in terms of their *precision-recall curves* (created by varying the code length from 4-bit to 64-bit while fixing the Hamming ball radius at 1), for retrieving original nearest neighbours and same-topic documents respectively[12]. It is clear that on all datasets and under both evaluation methodologies, STH outperforms binarised-LSI, LCH, and the state-of-the-art technique SpH (that has already been shown to work much better than LSH [1], stacked-RBM[13] [34] and boosting-SSC [38]). Using 16-bit codes and Hamming ball radius 1, the performance improvements are all statistically significant (P value < 0.01) according to one-sided micro sign test (s-test) [49].

We think the superior performance of STH is due to two reasons:

- the binary codes produced by binarised-LapEig effectively preserve the semantic similarity structure while maximising the entropy of the hash table;
- the maximum-margin hyperplane produced by linear-SVM ensures high generalisation ability [36].

We have also examined the approximation errors accumulated in each step of STH (see Section 3.3). Our anatomy reveals that almost all approximation errors come from the dimensionality reduction step using LapEig. However, LapEig does work better than alternative methods (such as LSI) for this step in our experiments, and it is a well-known hard problem to accurately detect the (intrinsic) dimensionality of data or effectively reduce the dimensionality of data. The median-based binarisation and SVM-based out-of-sample extension both work perfectly incurring little approximation errors.

The proposed STH approach (using binarised-LapEig and linear-SVM) to semantic hashing is pretty fast: on an ordinary PC with Intel Pentium4 3.00GHz CPU and 2GB RAM, our Matlab implementation of 64-bit STH takes approximately 0.0165 second per document for training (which is about 10 times faster than SpH), and 0.0007 second per document for prediction.

5. CONCLUSIONS

The main contribution of this paper is a novel Self-Taught Hashing (STH) approach to semantic hashing for fast similarity search. By decomposing the problem of finding small codes for large data into two stages — unsupervised learning and supervised learning — we achieve great flexibility in choosing learning algorithms. Using binarised-LapEig for the first stage and linear-SVM for the second stage, STH significantly outperforms binarised-LSI, LCH, and the state-of-the-art technique SpH [48]. Since STH is a general learning framework, it is promising to achieve even higher effectiveness and efficiency if more powerful unsupervised or supervised learning algorithms can be employed.

We shall apply this technique to text mining tasks (such as automated text categorisation [37]) and content-based multimedia retrieval [28] in the near future. It would also be interesting to combine semantic hashing and distributed computing (e.g., [29]) to further improve the speed and scalability of similarity search.

Acknowledgements

We are grateful to Dr Xi Chen (Alberta) for his valuable discussion and the London Mathematical Society (LMS) for their support of this work (SC7-09/10-6). We would also like to thank the anonymous reviewers for their helpful comments.

6. REFERENCES

[1] A. Andoni and P. Indyk. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. In *Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 459–468, Berkeley, CA, USA, 2006.

[2] S. Baluja and M. Covell. Learning to hash: Forgiving hash functions and applications. *Data Mining and Knowledge Discovery (DMKD)*, 17(3):402–430, 2008.

[3] M. Belkin and P. Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Computation*, 15(6):1373–1396, 2003.

[4] S. Belongie, C. Fowlkes, F. Chung, and J. Malik. Spectral partitioning with indefinite kernels using the nystrom extension. In *Proceedings of the 7th European Conference on Computer Vision (ECCV)*, pages 531–542, Copenhagen, Denmark, 2002.

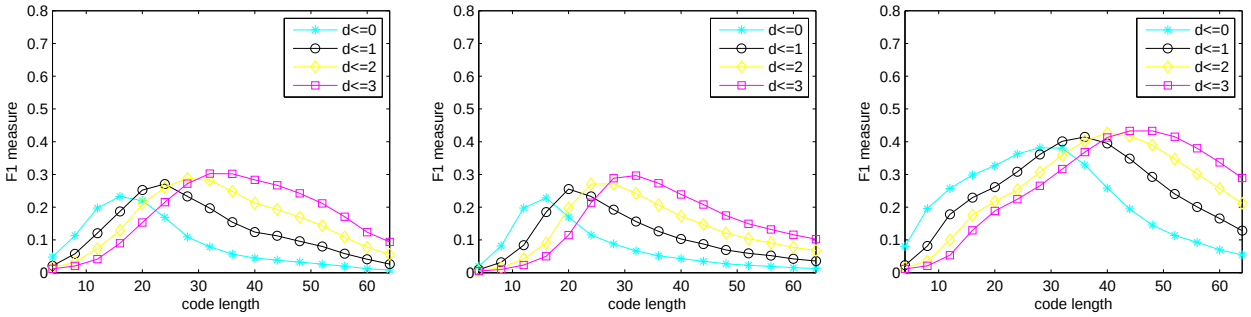
[5] M. W. Berry, S. T. Dumais, and G. W. O'Brien. Using linear algebra for intelligent information retrieval. *SIAM Review*, 37(4):573–595, 1995.

for their application of pattern recognition but obviously insufficient from the IR perspective. Due to this difference in performance measurement, their results are not directly comparable with ours.

[11] The F_1 measure scores reported here should not be directly compared with those in text categorisation papers, as we are addressing a very different problem even though the same datasets may have been used for experimentation.

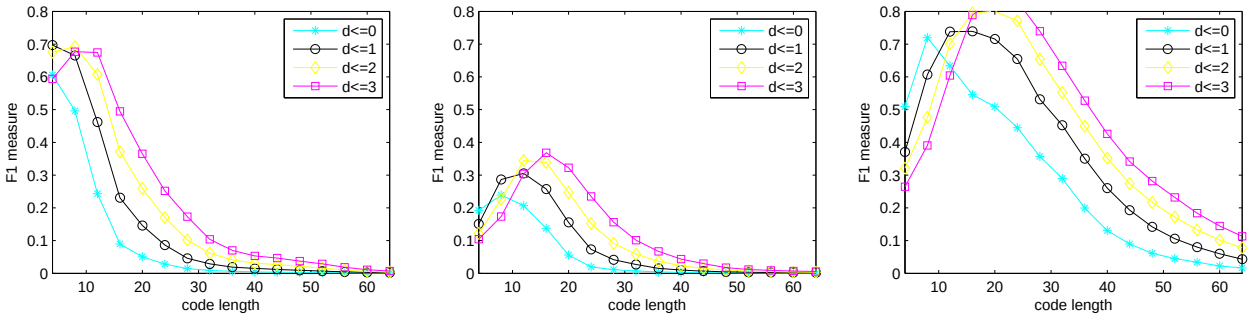
[12] Although we could achieve higher retrieval performance by utilising a bigger Hamming ball radius (e.g., 4), a large number of binary codes (e.g., $C_{64}^4 = 635376$ for 64-bit codes) would need to be checked for each query and then the efficiency gain brought by semantic hashing would diminish.

[13] For example, on the 20Newsgroups dataset, stacked-RBM achieves a maximum of $F_1 = 0.276$ for retrieving same-topic documents with 128-bit codes, while the same level of performance can be obtained using our STH approach with just 8-bit codes.



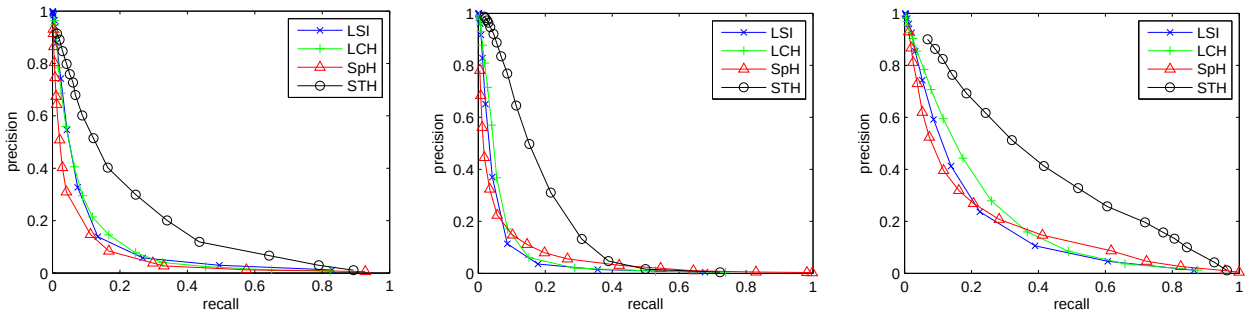
(a) Reuters21578 (b) 20Newsgroups (c) TDT2

Figure 2: The F_1 measure of STH for retrieving original nearest neighbours.



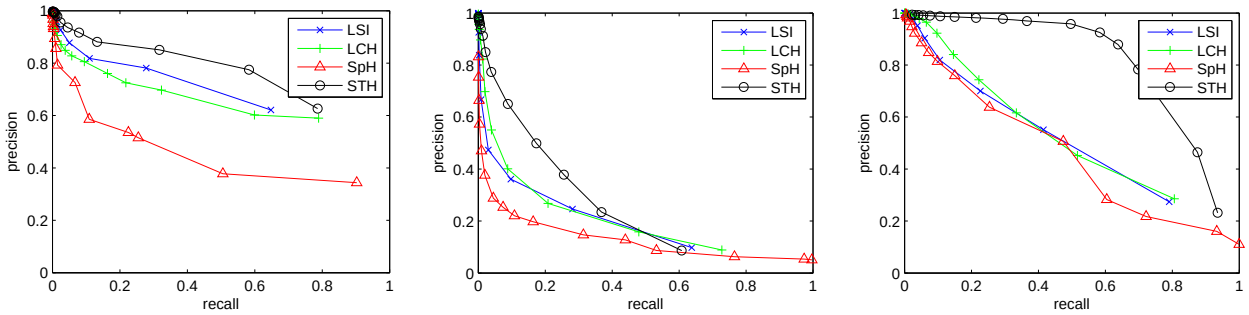
(a) Reuters21578 (b) 20Newsgroups (c) TDT2

Figure 3: The F_1 measure of STH for retrieving same-topic documents.



(a) Reuters21578 (b) 20Newsgroups (c) TDT2

Figure 4: The precision-recall curve for retrieving original nearest neighbours.



(a) Reuters21578 (b) 20Newsgroups (c) TDT2

Figure 5: The precision-recall curve for retrieving same-topic documents.

[6] F. R. K. Chung. *Spectral Graph Theory*. American Mathematical Society, 1997.

[7] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. MIT Press and McGraw-Hill, 2nd edition, 2001.

[8] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman. Indexing by latent semantic analysis. *Journal of the American Society of Information Science (JASIS)*, 41(6):391–407, 1990.

[9] P. Drineas and M. W. Mahoney. On the nystrom method for approximating a gram matrix for improved kernel-based learning. *Journal of Machine Learning Research (JMLR)*, 6:2153–2175, 2005.

[10] S. Dumais, J. Platt, D. Heckerman, and M. Sahami. Inductive learning algorithms and representations for text categorization. In *Proceedings of the 7th ACM International Conference on Information and Knowledge Management (CIKM)*, pages 148–155, Bethesda, MD, 1998.

[11] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin. LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874, 2008.

[12] G. H. Golub and C. F. V. Loan. *Matrix Computations*. The Johns Hopkins University Press, 3rd edition, 1996.

[13] L. W. Hagen and A. B. Kahng. New spectral methods for ratio cut partitioning and clustering. *IEEE Transactions on CAD of Integrated Circuits and Systems*, 11(9):1074–1085, 1992.

[14] A. Y. Halevy, P. Norvig, and F. Pereira. The unreasonable effectiveness of data. *IEEE Intelligent Systems*, 24(2):8–12, 2009.

[15] J. Hays and A. A. Efros. Scene completion using millions of photographs. *ACM Transactions on Graphics (TOG)*, 26(3):4, 2007.

[16] X. He, D. Cai, H. Liu, and W.-Y. Ma. Locality preserving indexing for document representation. In *Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pages 96–103, Sheffield, UK, 2004.

[17] X. He and P. Niyogi. Locality preserving projections. In *Advances in Neural Information Processing Systems (NIPS)*, volume 16, pages 153–160, Vancouver and Whistler, Canada, 2003.

[18] M. R. Henzinger. Finding near-duplicate web pages: A large-scale evaluation of algorithms. In *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pages 284–291, Seattle, WA, USA, 2006.

[19] G. Hinton, S. Osindero, and Y. W. Teh. A fast learning algorithm for deep belief nets. *Neural Computation*, 18(7):1527–1554, 2006.

[20] G. Hinton and R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, July 2006.

[21] C.-J. Hsieh, K.-W. Chang, C.-J. Lin, S. S. Keerthi, and S. Sundararajan. A dual coordinate descent method for large-scale linear svm. In *Proceedings of the 25th International Conference on Machine Learning (ICML)*, pages 408–415, Helsinki, Finland, 2008.

[22] T. Joachims. Text categorization with support vector machines: Learning with many relevant features. In *Proceedings of the 10th European Conference on Machine Learning (ECML)*, pages 137–142, Chemnitz, Germany, 1998.

[23] T. Joachims. *Learning to Classify Text using Support Vector Machines*. Kluwer, 2002.

[24] T. Joachims. Training linear SVMs in linear time. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 217–226, Philadelphia, PA, 2006.

[25] D. Knuth. *The Art of Computer Programming*. Addison-Wesley, 3rd edition, 1997.

[26] Y. Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 426–434, Las Vegas, NV, USA, 2008.

[27] K. Lang. Newsweeder: Learning to filter netnews. In *Proceedings of the 12th International Conference on Machine Learning (ICML)*, pages 331–339, Tahoe City, CA, 1995.

[28] M. S. Lew, N. Sebe, C. Djeraba, and R. Jain. Content-based multimedia information retrieval: State of the art and challenges. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMCCAP)*, 2(1):1–19, 2006.

[29] J. Lin. Brute force and indexed approaches to pairwise document similarity comparisons with mapreduce. In *Proceedings of the 32nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pages 155–162, Boston, MA, USA, 2009.

[30] C. D. Manning, P. Raghavan, and H. Schütze. *Introduction to Information Retrieval*. Cambridge University Press, 2008.

[31] T. Mitchell. *Machine Learning*. McGraw Hill, international edition, 1997.

[32] S. Pandey, A. Broder, F. Chierichetti, V. Josifovski, R. Kumar, and S. Vassilvitskii. Nearest-neighbor caching for content-match applications. In *Proceedings of the 18th International Conference on World Wide Web (WWW)*, pages 441–450, Madrid, Spain, 2009.

[33] R. Raina, A. Battle, H. Lee, B. Packer, and A. Y. Ng. Self-taught learning: Transfer learning from unlabeled data. In *Proceedings of the 24th International Conference on Machine Learning (ICML)*, pages 759–766, Corvalis, OR, USA, 2007.

[34] R. Salakhutdinov and G. Hinton. Semantic hashing. *International Journal of Approximate Reasoning (IJAR)*, 50(7):969–978, 2009.

[35] R. E. Schapire. The boosting approach to machine learning: An overview. In *Nonlinear Estimation and Classification*. Springer, 2003.

[36] B. Scholkopf and A. J. Smola. *Learning with Kernels*. MIT Press, Cambridge, MA, 2002.

[37] F. Sebastiani. Machine learning in automated text categorization. *ACM Computing Surveys*, 34(1):1–47, 2002.

[38] G. Shakhnarovich, P. A. Viola, and T. Darrell. Fast pose estimation with parameter-sensitive hashing. In *Proceedings of the 9th IEEE International Conference on Computer Vision (ICCV)*, pages 750–759, Nice, France, 2003.

[39] C. E. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27:379–423, 623–656, 1948.

[40] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 22(8):888–905, 2000.

[41] Q. Shi, J. Petterson, G. Dror, J. Langford, A. Smola, and S. Vishwanathan. Hash kernels for structured data. *Journal of Machine Learning Research (JMLR)*, 10:2615–2637, 2009.

[42] B. Stein. Principles of hash-based text retrieval. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pages 527–534, Amsterdam, The Netherlands, 2007.

[43] B. Stein, S. M. zu Eissen, and M. Potthast. Strategies for retrieving plagiarized documents. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pages 825–826, Amsterdam, The Netherlands, 2007.

[44] A. B. Torralba, R. Fergus, and Y. Weiss. Small codes and large image databases for recognition. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–8, Anchorage, AK, USA, 2008.

[45] R. Weber, H.-J. Schek, and S. Blott. A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces. In *Proceedings of 24th International Conference on Very Large Data Bases (VLDB)*, pages 194–205, New York City, USA, 1998.

[46] P. Wegner. A technique for counting ones in a binary computer. *Communications of the ACM (CACM)*, 3(5):322, 1960.

[47] K. Q. Weinberger, A. Dasgupta, J. Langford, A. J. Smola, and J. Attenberg. Feature hashing for large scale multitask learning. In *Proceedings of the 26th Annual International Conference on Machine Learning (ICML)*, page 140, Montreal, Quebec, Canada, 2009.

[48] Y. Weiss, A. B. Torralba, and R. Fergus. Spectral hashing. In *Advances in Neural Information Processing Systems (NIPS)*, volume 21, pages 1753–1760, Vancouver, Canada, 2008.

[49] Y. Yang and X. Liu. A re-examination of text categorization methods. In *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pages 42–49, Berkeley, CA, 1999.

[50] D. Zhang, J. Wang, D. Cai, and J. Lu. Laplacian co-hashing of terms and documents. In *Proceedings of the 32nd European Conference on IR Research (ECIR)*, page 577–580, Milton Keynes, UK, 2010.